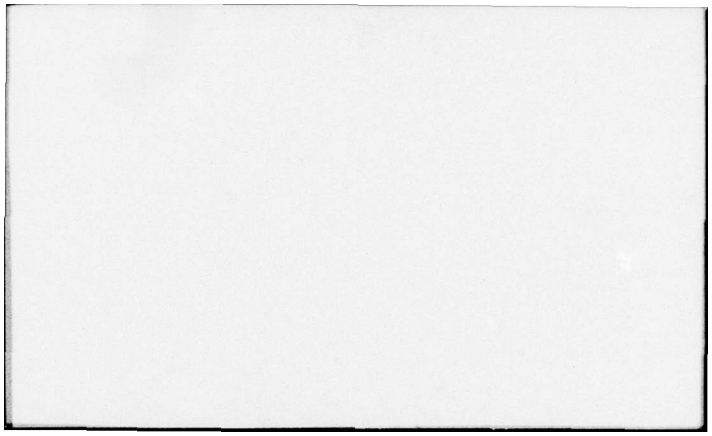
AD-A034 388 WHARTON SCHOOL OF FINANCE AND COMMERCE PHILADELPHIA P--ETC F/G 9/2
CONVERTING FROM RECTANGULAR TO RELATIONAL DATA BASE.(U)
SEP 76 D J ROOT N00014-75-C-0462
UNCLASSIFIED 76-09-09 NL

1 OF 1
AD
A034388

END

DATE
FILMED
2-77



CONVERTING FROM RECTANGULAR TO RELATIONAL DATA BASES

David J. Root

76-09-09

Prepared for the
Office of Naval Research
Information Systems
Arlington, Va. 22217
under
Contract N00014-75-C-0462
Project No. NR 049-272

Distribution Statement
Reproduction in whole or in part is permitted for any purpose of the United States Government

Department of Decision Sciences
The Wharton School
University of Pennsylvania
Philadelphia, Pennsylvania

DDC
RECEIVED
JAN 13 1977
A

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

letter on file
A

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: (14) 76-09-09
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): (6) CONVERTING FROM RECTANGULAR TO RELATIONAL DATA BASE.
5. TYPE OF REPORT & PERIOD COVERED: (9) Interim rept.
6. PERFORMING ORG. REPORT NUMBER: 76-09-09

7. AUTHOR(s): (10) David J. Root
8. CONTRACT OR GRANT NUMBER(s): N00014 0462 ~~N0014~~ 75-C-~~0464~~

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Department of Decision Sciences
The Wharton School of Finance & Commerce
University of Pennsylvania

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (15) N00014-75-C-0462 NR049-272

11. CONTROLLING OFFICE NAME AND ADDRESS
Office of Naval Research
Code 437, 800 North Quincy Avenue
Arlington, Va. 22217

12. REPORT DATE: (11) Sep[illegible]76
13. NUMBER OF PAGES: 22

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
(same) (12) 26p.

15. SECURITY CLASS. (of this report): Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)
Reproduction in whole or in part is permitted for any purposes of the United States' government

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
(same)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Relational Data Base
Data Conversion

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
This report describes a program developed for the conversion from rectangular to relational data bases. The programs intent is to facilitate loading some segment of a large rectangular file into a relational data base. Thus large rectangular files which could not easily be maintained as relational data bases could have some subet of interest loaded into a relational data base system, such as

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

408 757

20. (continued) Rel English, to take advantage of the more powerful relational query languages.

The inputs to the conversion routine are the subset of the rectangular file, basic information about the fields in the records, and prototype commands for loading the relational data base. The output is a file of commands for loading the records into the relational data base.

Section 1 Statement of Problem

In their paper Drs. O. Peter Buneman and Howard L. Morgan(1) outline the desirability of being able to restructure subsets of large rectangular data bases into relational data bases in order to make use of the flexibility of the interactive question-answering capabilities of existing relational data base languages. (This report uses REL English as an illustration of such a language.) The basic problem is that creating relational data bases is a very tedious task. For example, in REL English if it is desired to enter into the data base that John Smith is taking course IS 201 and that his major is Accounting, the following statements would be required:

```
JOHN SMITH:=NAME
IS 201:=NAME
ACCOUNTING:=NAME
STUDENT:=RELATION
MAJOR:=RELATION
JOHN SMITH IS A STUDENT OF IS 201.
ACCOUNTING IS THE MAJOR OF JOHN SMITH.
```

It is easy to see that if the information about a few

---

(1) "ASAP to REL: Efficient Relational Data Bases from Very Large Files", by Dr. O. Peter Buneman and Dr. Howard L. Morgan, Working Paper No. 75-01-06, Dept. of Decision Sciences, University of Pennsylvania.

p 02268

hundred students was to be entered by hand, it could be very time consuming. This information may already exist in a rectangular file, or it could be entered into one with considerably more ease than entering it into the relational data base. This report documents a program which given a rectangular file and a minimal amount of information about the desired statements (assertions) will generate the statements necessary to establish the required relational data base.

Section 2 Functional Design

A) Program Chain

Task: given a rectangular file

eg.

FIRST-NAME	LAST-NAME	COURSE	MAJOR
JOHN	SMITH	IS 201	ACCOUNTING

Output the assertions necessary to create a relational data base.

eg. (REL English)

STUDENT:=RELATION (1)

MAJOR:=RELATION (2)

JOHN SMITH:=NAME (3)

IS 201:=NAME (4)

ACCOUNTING:=NAME (5)

JOHN SMITH IS A STUDENT OF IS 201. (6)

ACCOUNTING IS THE MAJOR OF JOHN SMITH. (7)

Assertions 1 and 2 would not need to be repeated. Assertions such as 3, 4, 5, 6 and 7 would have to be repeated with new information from each record.

Therefore what would be necessary for the program would be a set of text indicating the necessary assertions.

eg.

```
STUDENT:=RELATION$
MAJOR:=RELATION$
&FIRST-NAME &LAST-NAME:=NAME$
&COURSE:=NAME$
&MAJOR:=NAME$
```

&FIRST-NAME &LAST-NAME IS A STUDENT OF &COURSE.$

&MAJOR IS THE MAJOR OF &FIRST-NAME &LAST-NAME.$

In this example of text, all references to record field names are preceded by an & (the variable delimiter) and each assertion is ended with a $ (the sentence delimiter). Both the variable delimiter and the sentence delimiter used in the text are specified in the input data.

The program will output all lines containing no variable delimiters. Each line containing a variable delimiter will be output once for each record, substituting the field values of that record for the field names in the text (the items preceded by a variable delimiter), deleting all leading and trailing blanks in the field value.

Besides the basic task of generating the assertions, two other features were thought desirable.

1) The program can be set to discard any generated assertions in which a blank field is placed or to substitute a given set of characters (possibly the null set) in place of the field. (Note for REL it was necessary to delete any assertions in which a totally blank field value was placed.)

2) The program can also be set to examine each character of the field, after leading and trailing blanks have been deleted, and sets of characters can be specified to be substituted for single characters. (eg. if a LISP based system were being used the blank character, " ", would have to be replaced with "/ ", i.e. "Mary Jane would have to be replaced with "Mary/ Jane").

The program would also require the location of the fields in the record.

eg.

FIELD NAME	BEGINNING COLUMN	ENDING COLUMN
FIRST-NAME	1	15
LAST-NAME	16	30
COURSE	31	35
MAJOR	36	50

Finally, the program would require the records for which the assertions are to be generated.

The basic operation of the program is as follows:

1) A set of code is generated which locates all nongenerating sentences (i.e. those containing no variable delimiters) in the text and which for all generating sentences (i.e. those containing variable delimiters) contains information about the location, in the record, of the field values referenced by the assertions, and about the sentence segments to be placed before or after each field value. Sentence segments are the portions of the genereating statements which are to be output in the assertions (eg. "IS A STUDENT OF").
2) Output all nongenerating sentences.
3) Given a record, use the code developed in step 1 to generate the required sentences and output them.
4) If duplicate assertions are to be deleted
   a) Sort all assertions into alphabetic order.

b) Delete all redundant assertions.

c) Resort remaining assertions into the order in which they were generated. (Note: the first assertion generated of multiply generated assertions is the one which is retained.)

b) Program usage

First, a rectangular file of the relevant subset of the data base is set up in the desired form. That is, all information stored in coded form is changed to the desired form for use in the relational data base language. (eg. Accounting may be stored as Acct.) Besides the form change, this step serves the purpose of isolating the desired subset of the rectangular data base.

Second, program CHAIN is run using the output from the first step, the prototype text, the field name directory, and punctuation information (see Section 3) as input.

The user will be prompted to indicate whether or not duplicate assertions are to be eliminated. The input text for program CHAIN must be in the order required by the relational data base language being used. (eg. In REL, data must be declared as a NAME before its relation to previously entered data can be asserted.) Therefore if the deletion of redundant assertions is required, it is accomplished by sorting the assertions first by the assertions themselves, subordered by order of generation number. A pass is then made through this new file, deleting all but the first assertion of each group of redundant

assertions. The remaining assertions are then resorted by their order of generation numbers and the final file is output without the order of generation numbers (as these are not needed in the input to the relational data base language).

Section 3 Operational Specifications

Four files are needed as input for the program.

1) The file of fixed length input records.

2) A directory file containing the length (number of ASCII characters) in an input record and the field locations (beginning and ending character numbers). The form of the file is

```
L
"F1"   B1   E1
"F2"   B2   E2
 .
 .
 .
```

where L is an integer equal to the number of characters (ASCII) in one input record. The Fi are the field names used in the text (these names must be quoted). The Bi are integers equal to the beginning character number of fields Fi. The Ei are integers equal to the ending character number of fields Fi. Each item in the file mus tbe separated by blank(s) and/or carriage return(s) from the other items.

3) The third file is the text of prototype assertions. where field values from the input records are to be substituted, the field names given in the field directory are used, preceded by a special character, the variable delimiter. Each individual assertion is ended with another special character, the sentence delimiter. These special

characters, are not included in the output assertions.

4) The punctuation file contains information about the special characters in the text and input records. The file is of the form

"V"

"S"

"P1" "P2" "P3" ...

"B" "A"

"C1" "R1"

"C2" "R2"

.

.

.

where V is the variable delimiter (a single character). S is the sentence delimiter (a single character). The Pi are single characters, which other than the sentence delimiter and variable delimiter would denote the end of a varibale name in the text. If B in the input is yes, all output assertions into which a blank field value is to be placed are discarded, if it is no they are kept. If B is no, then item A is to be included in the input file; if B is yes, then item A is not to be included in the input file. A is a set of characters which are to be put in the output assertions in place of the blank field values. The Ci are the single characters, which if they occur in a record field value are to be replaced by the character sets Ri before putting the field value in the output assertions. Each item

in the file is to be quoted and must be separated from the other items by blank(s) and/or carriage return(s). Also, if a quote mark, ", is to appear in any of the items, it must be input as two quote marks, "".

Once the program has started it will prompt the user for the device and file names of the input files. The record file is the only input file which does not have to be on a directory device (in which case it would not have a file name) as the other input files are all passed twice. If the user uses standard file names (i.e. REC.DAT for the record file, FLD.DAT for the field directory file, TEXT.DAT for the text file, and PUNC.DAT for the punctuation file) and they are all on disk in his area, he need not provide the device and file name for each file individually, but need only answer "yes" when asked if standard names and devices are used. If the user is providing the device and file names in answer to prompts and either a device or file name given is not available, the prompt will be given again until the answer given is a device or file name which is available. If it is desired to end such a cycle, the user must ^C back to monitor level (abort the job). The user will also be prompted for the device and file name (if a directory device is specified) for the output file. Finally, the user will be prompted to indicate whether or not duplicate assertions should be eliminated. If the system can handle duplicate assertions or the user knows that no duplicate assertions will be generated, indicating

that elimination of duplicate assertions is not necessary will save the sort-delete-resort steps.

In the course of execution the program will create and delete four temporary files: RECS1.TMP, RECS2.TMP, RECS3.TMP and RECS4.TMP. For sample input, output and prompting and answer runs see Appendix A1. Note that the assertions appear in the order given in the original text, and the first occurrence of a multiply generated assertion is the one that is retained, when duplicate assertions are eliminated.

Errors in the input information will most often cause an abort ending and the print out of a diagnostic statement. The diagnostic statements are

1) "FILE MUST BE ON A DIRECTORY DEVICE"

This is caused by specifying a device type for the text, punctuation, or field directory file which is not a directory device. This will not cause an abort ending, but will cause another request for the device type.

2) "VARIABLE DELIMITER OF MORE THAN ONE CHARACTER"

Variable delimiter specified in the punctuation file is of more than one character.

3) "SENTENCE DELIMITER OF MORE THAN ONE CHARACTER"

Sentence delimiter specified in the punctuation file is more than one character.

4) "PUNCTUATION OF MORE THAN ONE CHARACTER OR MISSPELLED BLANK DELETION "

One of the items in the punctuation file, which was

taken as a character denoting the end of a field name in the text, was more than one character.

5) "INSUFFICIENT INFORMATION IN PUNCTUATION FILE"

One of the required items was left out of the punctuation file.

6) "REPLACEMENT CHARACTER OF IMPROPER LENGTH OR BLANK SUBSTITUTE MISSING"

One of the items in the punctuation file, which was taken as a character to be replaced if found in a record field value, was more than one character.

7) "ATTEMPT TO READ OVER THE END OF FILE file name"

Could have been caused by leaving an item or quote mark out of the punctuation or field directory file, specifying an incorrect input record length, or leaving the final sentence delimiter out of the text file.

8) "IMPROPER COLUMN NUMBERS IN THE FIELD NAME DIRECTORY"

Cuased by a beginning or ending character number which is greater than the record length or less than 0, or a beginning character number which is greater than the ending character number given for that field.

9) "FIELD NAME DIRECTORY IS EMPTY"

No field names are given in the field directory file.

10) "IMPROPER INFORMATION IN THE FIELD DIRECTORY FILE"

Nonnumeric data given as character numbers in the field directory file

11) "NO GENERATING SENTENCES IN THE TEXT FILE"

None of the prototype assertions in the text file

contain variable delimiters.

12) "FIELD NAME name NOT FOUND IN FIELD DIRECTORY"

The given field name was used in the text, but was not found in the field directory.

Section 4 Bugs and Further Work

There are three notable problems with the current program. First at cerain points in the program in order to overcome a system bug, it was necessary to make a copy of character strings and process the copy, when processing the original would have been preferred

The second problem was that of reading in the character data. Other than reading in the characters one at a time (which is time consuming) or having the input quoted (a method used when not inconvenient), the only way to read in character data is to set a string up as the output channel and then perform a TRANSFILE. This function transfers all data (in 7 bit ASCII) from the input channel to the output channel until an end of file marker is encountered on either channel. This function is much more effiecient than reading in the characters one at a time, but if the end of file marker is encountered on the output channel first (i.e. the string is filled), when the next section of input is read in the last character which was attempted to be transferred to the string has been lost. Since on the current system records are separated by carriage returns, this was no problem. However, if the records were not separated by carriage returns, the character lost by using this method would be part of the next record, rather than part of the carriage return.

The third problem is that of the sorting routine used, a simple merge sort with no in core sorting. This routine

was used in hopes that at a later time some way might be found to hook up with the DEC 10 sorting routine. To date no method for doing this has been found, so the program still has its original and very inefficient (due to I/O) sorting routine.

Further improvements that might be desirable are

1) A more elaborate user interface such as the one illustrated in Dr.'s Morgan & Buneman's paper.

2) Have the program consult a dictionary for the information which the user provides in the field directory.

3) Allow substitution for character sets in the record field values rather than just single characters.

4) Allow the user to specify a record file and a set of logic conditionsto test the records with for inclusion in the assertion generation, rather than providing a previously screened input file.

5) Have the program do decoding of information in the rectangular file.

A copy of the current code (the program is written in ALGOL) is available from the Dept. of Decision Sciences at the University of Pennsylvania.

Appendix A1

The following are two sample runs. Each sample shows the input, then the prompting session and then the output. In the prompting sessions the computer typed portion is shown in capital letters and the user portion is shown in small letters (note: in actual usage the user answers could be in capital letters).

Input no. 1

Text (file: text.dat)

```
(RECTYPE PERSON NAME AGE RESIDENCE)%
(RECTYPE LOCATION CITY STATE)%
(GENCONS PERSON !NAME !AGE !CITY)%
(GENCONS LOCATION !CITY !STATE)%
```

Punctuation (file: punc.dat)

```
VARIABLE DELIMITER "!"
SENTENCE DELIMITER "%"
PUNCTUATION " "  ")"
DELETE ON BLANK FIELD "NO"  REPLACE WITH "UNDEF"
SUBSTITUTION CHARACTERS
" "  "/ "
```

Records (file: rec.dat)

```
JOHN                19BOSTON        MA
MARY JANE           22BALTIMORE     MD
ROBERT              24BOSTON        MA
JILL                21NEW YORK CITYNY
JEFFERY               PHILADELPHIA PA
```

Record field directory (file: fld.dat)

```
39
"NAME"   1   19
"AGE"   20   21
"CITY"  22   34
"STATE" 35   39
```

Prompting session 1

```
ARE ALL INPUT FILES ON DISK AND ARE THEY NAMED:
RECORD FILE = REC.DAT, TEXT FILE = TEXT.DAT
FIELD DIRECTORY FILE = FLD.DAT, PUNCTUATION FILE =
PUNC.DAT
ANSWER "YES" OR "NO", INCLUDE QUOTES
"yes"
ON WHAT DEVICE IS THE OUTPUT TO BE PLACED?
ANSWER: EG "DSK", INCLUDE QUOTES
"dsk"
WHAT IS THE NAME OF THE OUTPUT FILE TO BE?
ANSWER: EG "FIL.EXT", INCLUDE QUOTES
"finl.dat"
ARE DUPLICATE ASSERTIONS TO BE DELETED?
ANSWER "YES" OR "NO", INLUDE QUOTES
"yes"
```

Output no. 1 (file: finl.dat)

```
(RECTYPE PERSON NAME AGE RESIDENCE)
(RECTYPE LOCATION CITY STATE)
(GENCONS PERSON JOHN 19 BOSTON)
(GENCONS LOCATION BOSTON MA)
(GENCONS PERSON MARY/ JANE 22 BALTIMORE)
(GENCONS LOCATION BALTIMORE MD)
(GENCONS PERSON ROBERT 24 BOSTON)
(GENCONS PERSON JILL 21 NEW/ YORK/ CITY)
(GENCONS LOCATION NEW/ YORK/ CITY NY)
(GENCONS PERSON JEFFERY UNDEF PHILADELPHIA)
(GENCONS LOCATION PHILADELPHIA PA)
```

Input no. 2

Text (file: assert.dat)

```
STUDENT:=RELATION$
MAJOR:=RELATION$
&FIRST-NAME &LAST-NAME:=NAME$
&COURSE:=NAME$
&FIRST-NAME &LAST-NAME IS A STUDENT OF &COURSE.$
&MAJOR IS THE MAJOR OF &FIRST-NAME &LAST-NAME.$
```

Punctuation (file: puncl.dat)

```
VARIABLE DELIMITER "&"
SENTENCE DELIMITER "$"
PUNCTUATION ":" " " "."
DELETE ON BLANK FIELD "YES"
```

Records (file: record.dat)

```
JOHN            SMITH           IS201ACCOUNTING
TOM             JONES           IS201MATH
ARTHUR          BURNS           IS222FINANCE
```

Record field directory (file: field.dat)

```
50
"FIRST-NAME"  1  15
"LAST-NAME"  16  30
"COURSE"     31  35
"MAJOR"      36  50
```

```
Prompting session no. 2

     ARE ALL INPUT FILES ON DISK AND ARE THEY NAMED:
     RECORD FILE = REC.DAT, TEXT FILE = TEXT.DAT
     FIELD DIRECTORY FILE = FLD.DAT, PUNCTUATION FILE =
     PUNC.DAT
     ANSWER "YES" OR "NO", INCLUDE QUOTES
     "no"
     ON WHAT DEVICE IS THE PUNCTUATION FILE?
     ANSWER: EG "DSK", INCLUDE QUOTES
     "dsk"
     WHAT IS THE NAME OF THE PUNCTUATION FILE?
     ANSWER: EG "FIL.EXT", INCLUDE QUOTES
     "puncl.dat"
     ON WHAT DEVICE IS THE TEXT FILE?
     ANSWER: EG "DSK", INCLUDE QUOTES
     "dsk"
     WHAT IS THE NAME OF THE TEXT FILE?
     ANSWER: EG "FIL.EXT", INCLUDE QUOTES
     "assert.dat"
     ON WHAT DEVICE IS THE FIELD DIRECTORY FILE?
     ANSWER: EG "DSK", INCLUDE QUOTES
     "dsk"
     WHAT IS THE NAME OF THE FIELD DIRECTORY FILE?
     ANSWER: EG "FIL.EXT", INCLUDE QUOTES
     "field.dat"
     ON WHAT DEVICE IS THE RECORD FILE?
     ANSWER: EG "DSK", INCLUDE QUOTES
     "dsk"
     WHAT IS THE NAME OF THE RECORD FILE?
     ANSWER: EG "FIL.EXT", INCLUDE QUOTES
     "record.dat"
     ON WHAT DEVICE IS THE OUTPUT TO BE PLACED?
     ANSWER: EG "DSK", INCLUDE QUOTES
     "dsk"
     WHAT IS THE NAME OF THE OUTPUT FILE TO BE?
     ANSWER: EG "FIL.EXT", INCLUDE QUOTES
     "fin2.dat"
     ARE DUPLICATE ASSERTIONS TO BE DELETED?
     ANSWER "YES" OR "NO", INLUDE QUOTES
     "yes"
```

Output no. 2 (file: fin2.dat)

```
STUDENT:=RELATION
MAJOR:=RELATION
JOHN SMITH:=NAME
IS201:=NAME
JOHN SMITH IS A STUDENT OF IS201.
ACCOUNTING IS THE MAJOR OF JOHN SMITH.
TOM JONES:=NAME
TOM JONES IS A STUDENT OF IS201.
MATH IS THE MAJOR OF TOM JONES.
ARTHUR BURNS:=NAME
IS222:=NAME
ARTHUR BURNS IS A STUDENT OF IS222.
FINANCE IS THE MAJOR OF ARTHUR BURNS.
```

DISTRIBUTION LIST

Department of the Navy - Office of Naval Research

Data Base Management Systems Project

Defense Documentation Center
Cameron Station
Alexandria, VA 22314

Office of Naval Research
Code 102IP
Arlington, VA 22217

Office of Naval Research
Branch Office, Chicago
536 South Clark Street
Chicago, IL 60605

New York Area Office
715 Broadway - 5th Floor
New York, NY 10003

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
(Code RD-1)
Washington, DC 20380

Office of Naval Research
Code 458
Arlington, VA 22217

Assistant Chief for Technology
Office of Naval Research
Code 200
Arlington, VA 22217

Office of Naval Research
Information Systems Program
Code 437
Arlington, VA 22217

Office of Naval Research
Branch Office, Boston
495 Summer Street
Boston, MA 02210

Office of Naval Research
Branch Office, Pasadena
1030 East Green Street
Pasadena, CA 91106

Naval Research Laboratory
Technical Information Division
Code 2627
Washington, DC 20375

Office of Naval Research
Code 455
Arlington, VA 22217

Naval Electronics Laboratory Center
Advanced Software Technology Division
Code 5200
San Diego, CA 92152

Mr. E. H. Gleissner
Naval Ship Research and
Development Center
Computation & Mathematics Dept.
Bethesda, MD 20084

Mr. Kim B. Thompson
Technical Director
Information Systems Division
(OP-91T)
Office of Chief of Naval Operations
Washington, DC 20350

Professor Omar Wing
Columbia University
in the City of New York
Dept. of Electrical Engineering
and Computer Science
New York, NY 10027

Captain Grace M. Hopper
NAICOM/MIS Planning Branch
(OP-9160)
Office of Chief of Naval Operations
Washington, DC 20350

Bureau of Library and
Information Science Research
Rutgers - The State University
189 College Avenue
New Brunswick, NJ 08903
Attn: Dr. Henry Voos